State Machines Continued

Last time we
• Introduced the deterministic finite state machine
• Discussed the concept of state
• Talked about the parts and behaviors of state machines
• Found ways to define state machines (tables and diagrams)

Today we will
• Introduce the nondeterministic state machine
• Highlight differences between deterministic and nondeterministic machines
• Introduce ideas of equivalence between state machines: simulation and bisimulation

Facts About State Machines

• The state machines we consider in this class have (at least) one possible transition for each combination of current state and input. We will not encounter a situation where no transition is given for an input. Machines with this property are called receptive.
• For the state machines studied last lecture, there was exactly one possible transition for each combination of current state and input. These state machines are called deterministic.
• Sometimes it is useful to model a system using a state machine that may have more than one possible transition for each combination of current state and input. These state machines are called nondeterministic.
Nondeterministic State Machines

Nondeterministic state machines are like the deterministic state machines studied in the last lecture, except there may be more than one possible transition for a given current state and input.

When in state $s(n)=b$, if $x(n) = 0$, the next state $s(n+1)$ can be either $a$ or $b$.

The output $y(n)$ can be either $0$ or $1$.

In a nondeterministic state machine, there is more than one possible state response and output sequence.

Nondeterministic State Machines: Parts

Nondeterministic state machines are described with the same type of 5-tuple used for deterministic machines:

$$(\text{States}, \text{Inputs}, \text{Outputs}, \text{possibleUpdates}, \text{initialState})$$

The difference between the definitions is in the update function, now called $\text{possibleUpdates}$.

For a given input $x(n)$ and current state $s(n)$, $\text{possibleUpdates}$ provides the set of possible next states $s(n+1)$ and outputs $y(n)$.

$\text{possibleUpdates}: \text{States} \times \text{Inputs} \rightarrow \mathcal{P}(\text{States} \times \text{Outputs})$

Here, the notation $\mathcal{P}()$ denotes the power set. For some set $S$, $\mathcal{P}(S)$ is the set of all subsets of $S$.

Thus, the range of $\text{possibleUpdates}$ is a set of ordered pairs.
**Nondeterministic State Machines: Example**

Provide the 5-tuple definition for the following state machine:

\[
\begin{array}{c}
0/0 \quad 1/1 \quad \{0,1\}/1 \\
\end{array}
\]

States = \{a, b\}

Inputs = \{0, 1, absent\}

Outputs = \{0, 1, absent\}

initialState = a

\[
\begin{array}{|c|c|}
\hline
(s(n+1), y(n)) = possibleUpdates(s(n), x(n)) \\
\hline
x(n) = 0 & x(n) = 1 \\
\hline
\end{array}
\]

This example is a deterministic model of a parking meter.

The inputs are coin5 (5 minutes) coin25 (25 minutes) and tick.

The outputs are safe and expired.

The states represent the number of minutes left until expiration.

This model of meter operation is accurate, but complicated!
Here is a nondeterministic model of the parking meter.
It contains less detail – it is an abstraction of the previous model.
But, some outputs are possible in this model that are impossible
in the previous model:
x = coin5, tick, tick  can result in  y = safe, safe, expired
The meter expires 2 minutes after inserting a 5 minute coin!

State Machines:  Behaviors

A behavior of a state machine is a pair (x, y) where x is an input
sequence and y is a corresponding output sequence.  We define

\[
\text{Behaviors} = \{(x, y) \in [\text{Naturals}_0 \rightarrow \text{Inputs}] \times [\text{Naturals}_0 \rightarrow \text{Outputs}] \mid y \text{ is a possible output sequence for the input } x\}
\]

For a deterministic state machine, there is one output sequence y
for each input sequence x.  \text{Behaviors} is the graph of a function in
this case; each element in the domain \([\text{Naturals}_0 \rightarrow \text{Inputs}]\) is
mapped to one element in \([\text{Naturals}_0 \rightarrow \text{Outputs}]\).

For a nondeterministic state machine, there may be more than
one output sequence in \([\text{Naturals}_0 \rightarrow \text{Outputs}]\) corresponding to
each input sequence in \([\text{Naturals}_0 \rightarrow \text{Inputs}]\).  \text{Behaviors} is not the
graph of a function in this case, as a function cannot map a
domain element to more than one range element.
Equivalence of State Machines

Two different state machines may be equivalent in the sense that, given the same input sequence, they produce the same output sequence.

We identify two equivalence relations: simulation & bisimulation.

Roughly speaking, we say that machine A simulates machine B if, for any input sequence, every output of machine B is also a possible output of machine A.

We say that A bisimulates B if A simulates B and B simulates A.

Simulation Relations

A simulation relation associates the sets of two machines. It is a set of ordered pairs which match a state from machine A with an “equivalent” state from machine B.

Formally, we say that A simulates B if there exists a simulation relation \( S \subseteq States_B \times States_A \) such that

1. \((\text{initialState}_B, \text{initialState}_A) \in S\), and
2. \(\forall x(n) \in \text{Inputs,} \forall (s_B(n), s_A(n)) \in S\), and
   \(\forall (s_B(n+1), y_B(n)) \in \text{possibleUpdates}(s_B(n), x(n))\), and
   \(\exists (s_A(n+1), y_A(n)) \in \text{possibleUpdates}(s_A(n), x(n))\) such that
   \((s_B(n+1), s_A(n+1)) \in S \) and \(y_A(n) = y_B(n)\).
Simulation Relations

In plain words, A simulates B when there exists a set of state pairings so that any input $x(n)$ moves both machines from equivalent states $(s_B(n), s_A(n))$ to equivalent states $(s_B(n+1), s_A(n+1))$ generating the same output $y_A(n) = y_B(n)$.

If A simulates B, then A has all of the behaviors that B has, and maybe more.

$A$ simulates $B \Rightarrow \text{Behaviors}_B \subseteq \text{Behaviors}_A$

If A simulates B, then any behavior which is impossible for A, is also impossible for B.

Suppose A simulates B.
Then $(x, y) \notin \text{Behaviors}_A \Rightarrow (x, y) \notin \text{Behaviors}_B$

Simulation Relations: Example

$Inputs = \{1, \text{absent}\}, \quad Outputs = \{0, 1, \text{absent}\}$

Find a bisimilar state machine with two states, and give the bisimulation relation.